In [70]:
```
import pandas as pd
```

In [71]:
```
df = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data",names=["class","Alcohol","Malic acid","Ash","Alcalinity of ash","Magnesium","Total phenols","Flavanoids","Nonflavanoid phenols","Proanthocyanins","Color intensity","Hue","OD280/OD315 of diluted wines","Proline"])
```

In [72]:
```
df
```

Out[72]:

	class	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proli
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.640000	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.380000	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.680000	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.800000	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.320000	1.04	2.93	735
5	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.750000	1.05	2.85	1450
6	1	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30	1.98	5.250000	1.02	3.58	1290
7	1	14.06	2.15	2.61	17.6	121	2.60	2.51	0.31	1.25	5.050000	1.06	3.58	1295
8	1	14.83	1.64	2.17	14.0	97	2.80	2.98	0.29	1.98	5.200000	1.08	2.85	1045
9	1	13.86	1.35	2.27	16.0	98	2.98	3.15	0.22	1.85	7.220000	1.01	3.55	1045
10	1	14.10	2.16	2.30	18.0	105	2.95	3.32	0.22	2.38	5.750000	1.25	3.17	1510
11	1	14.12	1.48	2.32	16.8	95	2.20	2.43	0.26	1.57	5.000000	1.17	2.82	1280
12	1	13.75	1.73	2.41	16.0	89	2.60	2.76	0.29	1.81	5.600000	1.15	2.90	1320
13	1	14.75	1.73	2.39	11.4	91	3.10	3.69	0.43	2.81	5.400000	1.25	2.73	1150
14	1	14.38	1.87	2.38	12.0	102	3.30	3.64	0.29	2.96	7.500000	1.20	3.00	1547
15	1	13.63	1.81	2.70	17.2	112	2.85	2.91	0.30	1.46	7.300000	1.28	2.88	1310
16	1	14.30	1.92	2.72	20.0	120	2.80	3.14	0.33	1.97	6.200000	1.07	2.65	1280
17	1	13.83	1.57	2.62	20.0	115	2.95	3.40	0.40	1.72	6.600000	1.13	2.57	1130
18	1	14.19	1.59	2.48	16.5	108	3.30	3.93	0.32	1.86	8.700000	1.23	2.82	1680
19	1	13.64	3.10	2.56	15.2	116	2.70	3.03	0.17	1.66	5.100000	0.96	3.36	845
20	1	14.06	1.63	2.28	16.0	126	3.00	3.17	0.24	2.10	5.650000	1.09	3.71	780
21	1	12.93	3.80	2.65	18.6	102	2.41	2.41	0.25	1.98	4.500000	1.03	3.52	770
22	1	13.71	1.86	2.36	16.6	101	2.61	2.88	0.27	1.69	3.800000	1.11	4.00	1035
23	1	12.85	1.60	2.52	17.8	95	2.48	2.37	0.26	1.46	3.930000	1.09	3.63	1015
24	1	13.50	1.81	2.61	20.0	96	2.53	2.61	0.28	1.66	3.520000	1.12	3.82	845
25	1	13.05	2.05	3.22	25.0	124	2.63	2.68	0.47	1.92	3.580000	1.13	3.20	830
26	1	13.39	1.77	2.62	16.1	93	2.85	2.94	0.34	1.45	4.800000	0.92	3.22	1195
27	1	13.30	1.72	2.14	17.0	94	2.40	2.19	0.27	1.35	3.950000	1.02	2.77	1285
28	1	13.87	1.90	2.80	19.4	107	2.95	2.97	0.37	1.76	4.500000	1.25	3.40	915
29	1	14.02	1.68	2.21	16.0	96	2.65	2.33	0.26	1.98	4.700000	1.04	3.59	1035
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

148	3	13.32	3.24	2.38	21.5	92	1.93	0.76	0.45	1.25	8.420000	0.55	1.62	650
149	3	13.08	3.90	2.36	21.5	113	1.41	1.39	0.34	1.14	9.400000	0.57	1.33	550
150	3	13.50	3.12	2.62	24.0	123	1.40	1.57	0.22	1.25	8.600000	0.59	1.30	500
151	3	12.79	2.67	2.48	22.0	112	1.48	1.36	0.24	1.26	10.800000	0.48	1.47	480
152	3	13.11	1.90	2.75	25.5	116	2.20	1.28	0.26	1.56	7.100000	0.61	1.33	425
153	3	13.23	3.30	2.28	18.5	98	1.80	0.83	0.61	1.87	10.520000	0.56	1.51	675
154	3	12.58	1.29	2.10	20.0	103	1.48	0.58	0.53	1.40	7.600000	0.58	1.55	640
155	3	13.17	5.19	2.32	22.0	93	1.74	0.63	0.61	1.55	7.900000	0.60	1.48	725
156	3	13.84	4.12	2.38	19.5	89	1.80	0.83	0.48	1.56	9.010000	0.57	1.64	480
157	3	12.45	3.03	2.64	27.0	97	1.90	0.58	0.63	1.14	7.500000	0.67	1.73	880
158	3	14.34	1.68	2.70	25.0	98	2.80	1.31	0.53	2.70	13.000000	0.57	1.96	660
159	3	13.48	1.67	2.64	22.5	89	2.60	1.10	0.52	2.29	11.750000	0.57	1.78	620
160	3	12.36	3.83	2.38	21.0	88	2.30	0.92	0.50	1.04	7.650000	0.56	1.58	520
161	3	13.69	3.26	2.54	20.0	107	1.83	0.56	0.50	0.80	5.880000	0.96	1.82	680
162	3	12.85	3.27	2.58	22.0	106	1.65	0.60	0.60	0.96	5.580000	0.87	2.11	570
163	3	12.96	3.45	2.35	18.5	106	1.39	0.70	0.40	0.94	5.280000	0.68	1.75	675
164	3	13.78	2.76	2.30	22.0	90	1.35	0.68	0.41	1.03	9.580000	0.70	1.68	615
165	3	13.73	4.36	2.26	22.5	88	1.28	0.47	0.52	1.15	6.620000	0.78	1.75	520
166	3	13.45	3.70	2.60	23.0	111	1.70	0.92	0.43	1.46	10.680000	0.85	1.56	695
167	3	12.82	3.37	2.30	19.5	88	1.48	0.66	0.40	0.97	10.260000	0.72	1.75	685
168	3	13.58	2.58	2.69	24.5	105	1.55	0.84	0.39	1.54	8.660000	0.74	1.80	750
169	3	13.40	4.60	2.86	25.0	112	1.98	0.96	0.27	1.11	8.500000	0.67	1.92	630
170	3	12.20	3.03	2.32	19.0	96	1.25	0.49	0.40	0.73	5.500000	0.66	1.83	510
171	3	12.77	2.39	2.28	19.5	86	1.39	0.51	0.48	0.64	9.899999	0.57	1.63	470
172	3	14.16	2.51	2.48	20.0	91	1.68	0.70	0.44	1.24	9.700000	0.62	1.71	660
173	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.700000	0.64	1.74	740
174	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.300000	0.70	1.56	750
175	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.200000	0.59	1.56	835
176	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.300000	0.60	1.62	840
177	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.200000	0.61	1.60	560

178 rows × 14 columns

In [73]:
```
X = df[['Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids','Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']]
```

In [74]:
```
y = df[['class']]
```

In [75]:
```
from sklearn.model_selection import train_test_split
```

In [76]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.27)
```

In [77]:
```
from sklearn.preprocessing import StandardScaler
```

In [78]:
```
scaler = StandardScaler()
```

In [79]:
```
scaler.fit(X_train)
```

Out[79]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [80]:
```
X_train = scaler.transform(X_train)
```

In [81]:
```
X_test = scaler.transform(X_test)
```

In [82]:
```
X_train[1]
```

Out[82]:
```
array([-0.74524755, -0.65251605, -0.61782739,  0.86754863,  0.65760299,
       -0.44052224,  0.08015851, -0.25686738,  0.0684379 , -1.29556182,
        0.42111189,  0.55522778, -1.22325078])
```

In [83]:
```
from sklearn.neural_network import MLPClassifier
```

In [84]:
```
mlp = MLPClassifier(solver='adam', alpha=1e-5,hidden_layer_sizes=(30,30,30),max_iter=5000,random_state=1)
```

In [85]:
```
mlp.fit(X_train,y_train)
```

```
C:\Users\kavit\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:904: DataConversionWarning: A column
-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

apps

Out[85]:
```
MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(30, 30, 30), learning_rate='constant',
       learning_rate_init=0.001, max_iter=5000, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)
```

In [86]:
```
predictions = mlp.predict(X_test)
```

In [87]:
```
predictions
```

Out[87]:
```
array([2, 3, 3, 3, 3, 1, 3, 1, 3, 1, 2, 2, 3, 2, 1, 1, 2, 2, 3, 1, 2, 1, 1,
       1, 2, 3, 2, 1, 2, 2, 1, 2, 2, 2, 1, 1, 2, 1, 3, 2, 1, 1, 2, 1, 1, 3,
       3, 2, 3], dtype=int64)
```

In [88]:
```
y_test
```

Out[88]:

	class
67	2
162	3
138	3
152	3
133	3
1	1
144	3

21	1
70	2
20	1
82	2
91	2
153	3
124	2
41	1
56	1
98	2
128	2
147	3
23	1
65	2
50	1
49	1
33	1
120	2
146	3
74	2

30	1
118	2
61	2
55	1
116	2
89	2
103	2
7	1
8	1
95	2
9	1
140	3
109	2
54	1
19	1
93	2
24	1
17	1
96	2
177	3

96	2
177	3
79	2
166	3

In [89]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [90]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[18  0  0]
 [ 0 18  2]
 [ 0  0 11]]
```

In [91]:
```
print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

          1       1.00      1.00      1.00        18
          2       1.00      0.90      0.95        20
          3       0.85      1.00      0.92        11

avg / total       0.97      0.96      0.96        49
```